REMARKS

The Examiner has rejected Claims 1-3, 12-18, 27-30, and 91-92 under 35 U.S.C. 101 as being directed toward non-statutory subject matter. Applicant has clarified the claims to include a computer program product "embodied on a tangible computer readable medium" in order to avoid such rejection.

The Examiner has objected to Claim 29 due to an informality. Such objection is deemed moot in view of the correction of Claim 29 hereinabove.

The Examiner has further rejected each of the claims on the grounds of non-statutory obviousness-type double patenting as being unpatentable over Claims 1-21 of U.S. Patent No. 7,107,574. Applicant asserts that such rejection is overcome in view of the filing of the terminal disclaimer submitted herewith.

The Examiner has rejected Claims 1-3, 12-18, 27-33, 42-48, 57-63, 72-78, and 87-92 under 35 U.S.C. 102(e) as being anticipated by Harvey et al. (U.S. Patent Publication No. 2006/0190575). In addition, the Examiner has rejected Claims 1-3, 12-18, 27-33, 42-48, 57-63, 72-78, and 87-92 under 35 U.S.C. 103(a) as being unpatentable over Uszok et al. (U.S. Publication No. 2004/0205772) in view of Kouznetsov et al. (U.S. Patent No. 6,931,546). Applicant respectfully disagrees with such rejections.

With respect to the 35 U.S.C. 102(e) rejection of the independent claims, the Examiner has relied on the following excerpts from the Harvey reference to make a prior art showing of applicant's claimed "matching code to match each complex data type with an associated execution process available to said destination computer" (see this or similar, but not necessarily identical language in the independent claims).

> "[0055] In response, based on the device's unique identifier, **Configuration Server 116 retrieves one of the configuration templates 210 that is associated with the device 114. The retrieved template is parameterized.** Parameter values are resolved by retrieving specific values from Directory 110, or another repository, that correspond to the parameters. As a

result, **a stream or file of XML format configuration information is created and potentially stored.** The XML information comprises one or more XML tags and attributes that contain one or more CLI commands that represent a new device configuration. Each element of general device configuration information is embedded in a <config> XML tag and each device CLI command is embedded in a <cli> XML tag. The specific tags set forth herein are not required; functionally equivalent tags having different names may be used." (emphasis added)

"[0066] In push mode, **configuration data can be pushed to the device 114 in the form of an event.** For example, device 114 subscribes to a configuration announcement event that is managed by event service 120. In this arrangement, XML format configuration information is sent as a payload of an event using the Event Gateway 118. Push mode may be used to send identical configuration information to multiple devices, akin to a broadcast." (emphasis added)

"[0067] In pull mode, **Configuration Agent 200 pulls event information from a Web server** using an HTTP GET request on receipt of the event. For example, a signal event is sent to trigger the device 114 to obtain an incremental configuration from Configuration Server 116. This may be used when, in response to a particular event, it is desirable to cause the device to load a new or updated configuration, or when identical information cannot be sent to multiple devices, e.g., due to differences in the content of the information." (emphasis added)

Applicant respectfully points out that the above excerpts relied on by the Examiner merely teach that the "Configuration Server 116 retrieves one of the configuration templates 210 that is associated with the device," that the "retrieved template is parameterized," and that "a stream or file of XML format configuration information is created and potentially stored." Additionally, the above reference citations teach that "configuration data can be pushed to the device 114 in the form of an event" and that "Configuration Agent 200 pulls event information from a Web server."

However, the retrieval and parameterization of a configuration template, the creation and storage of XML configuration information, and the pushing and pulling of configuration data does not even suggest "matching code to match each complex data type with an associated execution process" (emphasis added), as claimed by applicant.

Additionally, the Harvey reference teaches that "[w]eb server 206 delivers XML encoded configuration information from the server to the Configuration Agent 200"

(paragraph [0058]) and that "Configuration Agent 200 carries out a parsing and checking process on the received XML configuration information" in order to "[determine] whether the XML information is well-formed" and "verify that it has received all the XML configuration information" (paragraph [0059]), as well as to perform a "CLI syntax check" (paragraph [0060]). However, parsing in order to check the received XML configuration and syntax, as in Harvey, does not teach "matching code to match each complex data type with an associated execution process available to said destination computer" (emphasis added), as claimed by applicant.

With respect to the independent claims, the Examiner has further relied on the following excerpts from the Harvey reference to make a prior art showing of applicant's claimed "triggering code to trigger processing by the or each execution process associated with a complex data type within said operation specifying XML data" (see this or similar, but not necessarily identical language in the independent claims).

> "[0054] When the device 114 is powered-up at customer premises 112, the device connects to Configuration Server 116 by establishing a TCP/IP connection. The device **issues an HTTP get request to Web server 206 of Configuration Server 116.** To uniquely identify itself to the Configuration Server 116 and to the Web server, the **device 114 provides its token** as a unique identifier." (emphasis added)
>
> "[0064] --Partial Configuration"

Applicant respectfully notes that the above excerpts merely teach that the "device issues an HTTP get request to Web server 206 of Configuration Server 116" and "provides its token as a unique identifier." Issuing an "HTTP get request" and sending a "token as a unique identifier," as in Harvey, fails to even suggest "triggering code to trigger processing by the or each execution process associated with a complex data type within said operation specifying XML data" (emphasis added), as claimed by applicant.

With respect to the independent claims, the Examiner has further relied on the following excerpt from the Harvey reference to make a prior art showing of applicant's

claimed technique "wherein said execution process maps configuration data specified within said operation specifying XML data to a configuration data store of said destination computer; wherein said configuration data store is one of: a Windows Registry entry; an INI file; a DAPI store; and a database entry" (see this or similar, but not necessarily identical language in the independent claims).

> "[0049] The network device 114 arrives at the customer premises 112. A cable installer visits the customer premises 112, installs cable and plugs in the network device 114. The cable installer does not have any expert knowledge of the network device 114. The device 114 is preprogrammed with or discovers a token that uniquely identifies itself, e.g., a router hostname or MAC address. Upon power up, the device 114 **establishes a connection to Configuration Server 116.** The device 114 identifies itself, using the token, to the server and **requests the configuration information** that is held in Directory 110 for the device." (emphasis added)

Applicant respectfully points out that the above excerpt merely discloses that "the device 114 establishes a connection to Configuration Server 116" and that the device "requests the configuration information that is held in Directory 110 for the device." However, "establish[ing] a connection to [the] Configuration Server" and "request[ing]... configuration information," as in Harvey, does not teach mapping "configuration data specified within said operation specifying XML data to a configuration data store," much less a technique "wherein said execution process maps configuration data specified within said operation specifying XML data to a configuration data store of said destination computer, wherein said configuration data store is one of: a Windows Registry entry; an INI file; a DAPI store; and a database entry" (emphasis added), as claimed by applicant.

With respect to the independent claims, the Examiner has relied on paragraph [0055] (reproduced above), in addition to the following excerpt from the Harvey reference, to make a prior art showing of applicant's claimed technique "wherein said operation includes returning result data from said destination computer to said source computer in dependence upon said operation performed by said execution process," as well as applicant's claimed technique "wherein said result data includes data specifying

existing configuration data of said destination computer" and "wherein said execution process maps existing configuration data of said destination computer stored within said configuration data store of said destination computer to said result data to be returned to said source computer" (see this or similar, but not necessarily identical language in the independent claims).

> "[0056] Configuration Server 116 then **checks whether the XML configuration information is valid** in comparison to a specified Document Type Definition (DTD) that defines a grammar with which the XML configuration information must conform. This task may be carried out using a conventional XML parser, or an XML validation parser, of which several are commercially available. Such checking may also involve testing whether the XML configuration information constitutes well-formed XML text, and syntactically correct XML text. Determination of whether the XML information is well-formed may involve comparing the configuration information to the DTD, determining whether all opening tags have corresponding closing tags such that there is no nesting, etc. If the XML is not well-formed, an error is generated. Syntax checking at this stage involves **checking the syntax of the XML text**, without determining whether CLI strings contained within XML tags of the XML text constitute syntactically correct CLI commands." (emphasis added)

As mentioned above, applicant respectfully points out that the excerpts relied on by the Examiner merely teach that the "Configuration Server 116 retrieves one of the configuration templates 210 that is associated with the device," that the "retrieved template is parameterized," and that "a stream or file of XML format configuration information is created and potentially stored" (paragraph [0055]). In addition, the excerpts also teach "checking the syntax of the XML text" in order to "[check] whether the XML configuration information is valid in comparison to a specified Document Type Definition (DTD)."

However, the mere disclosure of retrieving and parameterizing a configuration template at the Configuration Server, in addition to creating and storing XML configuration information, and checking the syntax of XML text, as in Harvey, simply fails to disclose "returning result data from said destination computer," much less a technique "wherein said operation includes returning result data from said destination computer to said source computer in dependence upon said operation performed by said

execution process," or a technique "wherein said result data includes data specifying existing configuration data of said destination computer" and "wherein said execution process maps existing configuration data of said destination computer stored within said configuration data store of said destination computer to said result data to be returned to said source computer" (emphasis added), as claimed by applicant. Clearly, retrieving a configuration template at the Configuration Server and storing configuration information, as in Harvey, simply fails to even suggest "returning result data," in the manner as claimed by applicant.

With respect to the independent claims, the Examiner has relied on paragraphs [0055] and [0056] (cited above), in addition to the following excerpt from the Harvey reference, to make a prior art showing of applicant's claimed technique "wherein said operation specifying XML data is parsed after validating said operation specifying XML data to extract at least one identifier for mapping said at least one identifier to an available execution process" (see this or similar, but not necessarily identical language in the independent claims).

> "[0051] Upon receipt of a configuration file for the device 114 from the Configuration Server 116, the device **validates the configuration file** by doing a **syntax check.** In this context, a "syntax check" may involve one or more separate checks, including a check of whether the template is well-formed; validation of the configuration file; and a syntax check of one or more native commands that are contained in the configuration file. If the syntax check is successful, then the device 114 applies the configuration information and writes it to persistent storage within the device." (emphasis added)

As mentioned above, applicant respectfully points out that the excerpts from Harvey relied on by the Examiner merely teach that the "Configuration Server 116 retrieves one of the configuration templates 210 that is associated with the device," that the "retrieved template is parameterized," and that "a stream or file of XML format configuration information is created and potentially stored" (paragraph [0055]). In addition, Harvey also teaches "checking the syntax of the XML text" in order to "[check] whether the XML configuration information is valid in comparison to a specified Document Type Definition (DTD)" (paragraph [0056]). Further, Harvey teaches that

"the device validates the configuration file by doing a syntax check" which "may involve one or more separate checks, including a check of whether the template is well-formed, validation of the configuration file; and a syntax check of one or more native commands that are contained in the configuration file."

However, the mere disclosure of retrieving and parameterizing a configuration template, creating and storing XML configuration information, and checking the syntax of XML text by parsing the XML, in addition to validating the configuration file by doing a syntax check, as in Harvey, simply fails to disclose a technique "wherein said operation specifying XML data is parsed after validating said operation specifying XML data to extract at least one identifier for mapping said at least one identifier to an available execution process" (emphasis added), as claimed by applicant. Clearly, the mere disclosure of storing created XML configuration data, as in Harvey, fails to even suggest "extract[ing] at least one identifier for mapping said at least one identifier to an available execution process" (emphasis added), in the manner as claimed by applicant.

The Examiner is reminded that a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described in a single prior art reference. *Verdegaal Bros. v. Union Oil Co. Of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). Moreover, the identical invention must be shown in as complete detail as contained in the claim. *Richardson v. Suzuki Motor Co.*868 F.2d 1226, 1236, 9USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim.

This criterion has simply not been met by the above reference, as noted above. Nevertheless, despite such paramount deficiencies and in the spirit of expediting the prosecution of the present application, applicant has incorporated the subject matter former Claim 91 into the independent claims to further distinguish applicant's claim language from the above reference.

With respect to the subject matter of former Claim 91 (now at least substantially incorporated into the independent claims), the Examiner has relied on paragraphs [0051], [0055], and [0056] (cited above) from the Harvey reference to make a prior art showing of applicant's claimed "validating said operation specifying XML data received at said destination computer against schema data, where said schema data is sent to said destination computer from said source computer at the same time as said operation specifying XML data" (see this or similar, but not necessarily identical language in the independent claims).

Applicant respectfully points out that the excerpts from Harvey relied on by the Examiner merely teach that the "Configuration Server 116 retrieves one of the configuration templates 210 that is associated with the device" (paragraph [0055] – emphasis added). Further, Harvey teaches that "Configuration Server 116 then checks whether the XML configuration information is valid in comparison to a specified Document Type Definition (DTD) that defines a grammar with which the XML configuration information must conform" (paragraph [0056] – emphasis added).

However, the mere disclosure that the Configuration Server checks if the retrieved configuration template is valid in comparison to a specified Document Type Definition, as in Harvey, simply fails to even suggest "validating said operation specifying XML data received at said destination computer against schema data, where said schema data is sent to said destination computer from said source computer at the same time as said operation specifying XML data" (emphasis added), as claimed by applicant. Clearly, checking configuration information to a specified Document Type Definition, as in Harvey, fails to suggest that "said schema data is sent to said destination computer from said source computer at the same time as said operation specifying XML data" (emphasis added), in the manner as claimed by applicant.

Again, the foregoing anticipation criterion has simply not been met by the above reference, as noted above.

With respect to the 35 U.S.C. 103(a) rejection of the independent claims, the Examiner has relied on paragraphs 0014, 0050, and 0055 from the Uszok reference to make a prior art showing of applicant's claimed technique "receiving code to receive at said destination computer operation specifying XML data sent by said source computer" (see this or similar, but not necessarily identical language in the independent claims).

Applicant respectfully asserts that the excerpts from Uszok relied upon by the Examiner discloses a technique where "[t]he client side--mBot--implements a presentation layer, active component or even pure XML or HTML, and may have multiple presentations for different platforms" (Paragraph 0014 - emphasis added). However, having the client side implement pure XML simply, as in Uszok, fails to meet a "receiving code to receive at said destination computer operation specifying XML data sent by said source computer" (emphasis added), as claimed by applicant.

In the Office Action dated 03/08/2006, the Examiner has further argued that "the plug-ins can be passed from the botServer manager to plug-in manager, which is the target process" and that "[b]oth managers are performed at the same target computer botServer." Additionally, the Examiner cited the following excerpts from Uszok to support such assertion.

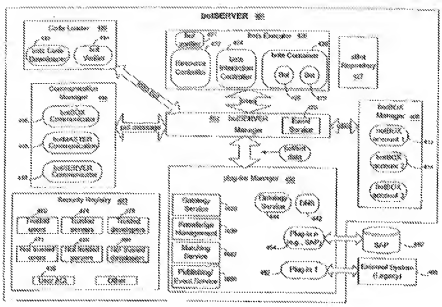

Figure 4

(Uszok, Figure 4)

"[0079] Plug-ins: The botServer standard services can be extended by a mechanism of plug-ins. **A plug-in is a software component installed by botServer administrator within the plug-ins manager (450) which communicates with other plug-ins and bots via the botServer manager 402 using messages.** From the usability point of view, a plug-in is somewhat similar to a bot--it has an unique ID, understands bot messages, participates in protocols etc. A plug-in, however, never migrates, cannot be automatically downloaded and installed, and usually has much higher system security permissions. A plug-in also may possess any number of system threads (since it doesn't have to strictly adhere to a request-response execution schema). Plug-ins are usually used to interface with various external systems, perform demanding real-time operations and provide other services. For example, FIG. 4 illustrates a plug-in 454 for interfacing with an SAP server 462. Another plug-in 452 interfaces with a legacy system 460 such as an enterprise database running on a mainframe. FIG. 17 shows auction and B2B plug-ins on botServers.

[0080] Plug-ins may be configured so that they have the same ID on every server. That strategy relieves bots from having to search for an appropriate plug-in if they know the requested plug-in ID. Some standard system services (e.g. Ontology service 1620 or botDNS 442) may be implemented as plug-ins as well. Specialized plug-ins can be created by botServer owners to provide services that make the server more attractive to bots (e.g. in order to advertise some other profitable services). Other examples shown in FIG. 4 (and FIG. 16) are knowledge management service 1630, matching service 1642 and a publishing/event service 1650. Generally the plug-ins should be created to be compatible with botServer scaling." (Uszok, Paragraphs 0079, 0080 - emphasis added)

Applicant respectfully asserts that the additional excerpts from Uszok relied upon by the Examiner simply disclose that "[a] plug-in is a software component installed by botServer administrator within the plug-ins manager (450) which communicates with other plug-ins and bots via the botServer manager 402 using messages" (emphasis added). However, the "get/set data" and "uses" flows that communicate with the plug-ins and bots "using messages" in Uszok's Figure 4 fails to disclose "receiving code to receive at said destination computer operation specifying XML data sent by said source computer" (emphasis added), as claimed by applicant. In addition, Uszok's disclosure that "[b]oth managers are performed at the same target computer botServer" (emphasis added) simply fails to meet "receiving code to receive at said destination computer operation specifying XML data sent by said source computer" (emphasis added), as claimed by applicant.

In the Office Action dated 09/22/2006, the Examiner reiterated their previous arguments and additionally asserted "Uszok states that data can be exchanged in the XML format... in the agent based system." Additionally, the Examiner cited the following excerpts from Uszok to support such assertions.

> "[0065] User Model data are acquired over time by "watching" the user's activities via the GUI 520. Recall that interfaces are provided to various user applications such word processing, email, and web browser. The user's actions are captured as data messages and routed by the botMaster core 530 to the Knowledge Manager component 590 for recording in the User Model 592. **Knowledge of the user**--the user model preferably is stored as a conceptual graph. It **may be imported from external sources, for example using RDF,** the Resource Description Framework. RDF is a framework for metadata under development by W3C to provide interoperability between applications that exchange machine-understandable information on the Web. See www.w3.org/rdf. **It can use XML as an interchange syntax."** (emphasis added)

Applicant respectfully notes that the above citation teaches that "the user model preferably is stored as a conceptual graph," which "may be imported from external sources, for example using RDF," which "can use XML as an interchange syntax." However, teaching that "a conceptual graph... may be imported from external sources [using] use XML as an interchange syntax" fails to disclose "receiving code to receive at

said destination computer operation specifying XML data sent by said source computer" (emphasis added), as claimed by applicant.

In addition, with respect to the independent claims, the Examiner has relied on the following excerpt from the Uszok reference to make a prior art showing of applicant's claimed "parsing code to parse said operation specifying XML data to identify one or more complex data types within said operation specifying XML data" and "matching code to match each complex data type with an associated execution process available to said destination computer" (see this or similar, but not necessarily identical language in the independent claims).

> "[0057] Returning now to the bot code acquisition process, the botServer 350 may already have a copy of the corresponding sBot class for operation with the mBot. (Indeed, the botServer 350 could even be hosted on the same site as the bot developer site 310.) **If so, the botServer 350 creates an instance of the requested sBot and assigns an identifier to it corresponding to the mBot that made the request.** If the appropriate sBot code is not already present, the botServer 350 obtains the appropriate sBot code from the bot developer site 310 by download link 362. Authentication of that code, security issues, payment and other particulars are discussed later in this description. **At the botServer, after the sBot is installed, a message originating from botMaster 320 is communicated to the sBot code to initialize it to carry out the task requested by the user.**" (Uszok, Paragraph 0057 - emphasis added).

Applicant respectfully asserts that such excerpt from Uszok relied upon by the Examiner merely teaches that the "botServer 350 creates an instance of the requested sBot and assigns an identifier to it corresponding to the mBot that made the request" (emphasis added). Additionally, Uszok discloses that "[a]t the botServer, after the sBot is installed, a message originating from botMaster 320 is communicated to the sBot code to initialize it to carry out the task requested by the user" (emphasis added). However, the mere disclosure of creating an sBot instance and initializing it from "a message originating from botMaster" simply, as in Uszok, fails to meet a technique of "parsing code to parse said operation specifying XML data to identify one or more complex data types within said operation specifying XML data" (emphasis added) and "matching code

to match each complex data type with an associated execution process available to said destination computer" (emphasis added), as claimed by applicant.

In the Office Action dated 03/08/2006, the Examiner has further argued that "Uszok states matching mechanism (0133) and the XML-based interaction protocols define a set of states that a bot may exist in, rules of transition from one state to another and a description of the schema of data that can be exchanged between participating parties in order to transition from one state to another (0128)." Additionally, the Examiner cited the following excerpts from Uszok to support such assertion.

> "[0128] A meeting place in the present system is a separated logical area of the botServer, where bots can interact with chosen types of other bots and plug-ins according to defined interaction protocols. Interaction protocols define a set of states that a bot may exist in, rules of transition from one state to another and [optionally] **a description of the structure (schema) of documents (data) that can be exchanged between participating parties in order to transition from one state to another. An interaction protocol can be described, for example, in an XML-based language.** The selected protocols are implemented on bot's and plug-ins in accordance with the present invention." (Uszok, Paragraph 0128 – emphasis added).
>
> '[0133] Matching is a general mechanism that enables selection of abstract "offers" that most closely match abstract "requests." The mechanism is abstract, because it does not define what the offer really is. **The determination is based only on its description and applying sophisticated AI algorithms to actually perform matching, however XML-based, "hard"--regular expression or SQL-like based, matching might be used as well** (whichever more useful in a given situation). Such an approach allows, e.g., selecting botServers providing most appropriate services, or selecting trade offers closest to the buyer's requirements--all using the same mechanism. Generally the matching service should be able to provide a match of request and offer even if they cannot be matched by simple comparison.' (Uszok, Paragraph 0133 – emphasis added).

Applicant respectfully asserts that the matching disclosed in the excerpts from Uszok relied upon by the Examiner teach that a 'determination is based only on its description and applying sophisticated AI algorithms to actually perform matching, however XML-based, "hard"--regular expression or SQL-like based, matching might be used as well' (emphasis added). However, generally disclosing XML-based matching, as

in Uszok, fails to meet "matching code to match each complex data type with an associated execution process available to said destination computer" (emphasis added), as claimed by applicant. Further, Uszok's disclosure where "an interaction protocol can be described, for example, in an XML-based language" simply fails to meet "parsing code to parse said operation specifying XML data to identify one or more complex data types within said operation specifying XML data" (emphasis added), as claimed by applicant.

In the Office Action dated 09/22/2006, the Examiner merely reiterated the Examiner's previous rejection of applicant's claimed language. Again, applicant respectfully notes that none of the language cited by the Examiner teaches "parsing code to parse said operation specifying XML data to identify one or more complex data types within said operation specifying XML data" and "matching code to match each complex data type with an associated execution process available to said destination computer," as specifically claimed by applicant.

Further, with respect to the independent claims, the Examiner has relied on the following excerpt from the Kouznetsov reference to make a prior art showing of applicant's claimed technique "wherein said execution process maps configuration data specified within said operation specifying XML data to a configuration data store of said destination computer; wherein said configuration data store is one of: a Windows Registry entry, an INI file; a DAPI store; and a database entry" (see this or similar, but not necessarily identical language in the independent claims).

> "The above limitations of the prior art are addressed by a system, method and software in which a process is run on a client machine having sufficient privileges to execute privileged processes. This process has a role of a "local system" and is effectively an administrator for the machine. An agent program running in user-mode provides a generic interface. **The agent includes an application programming interface ("API") for receiving requests for privileged processes.** The agent includes an interface to the privileged process as well. The agent includes methods for authenticating any received requests and will only forward a request to the privileged process upon determining that the requesting application has sufficient trust. Hence, the agent provides a level of indirection in accessing the privileged process so that the local system interface is not exposed directly to untrusted entities.

> Briefly stated, the present invention involves a system for providing application services in a computing environment having both user-mode processes and privileged-mode processes. An agent executes in privileged mode and exposes an interface to user-mode processes. A user-mode component is provided with an interface configured to access the agent's exposed interface. **A configuration component specifies a list of installable code components that are authorized for installation, wherein the agent will only execute privileged-mode functions in response to accesses by the user-mode code component when the installable code component is represented on the list.'** (Kouznetsov, Col. 4, lines 25-54 – emphasis added)

Applicant respectfully asserts that the above excerpt from Kouznetsov relied upon by the Examiner teaches that an 'agent includes an application programming interface ("API") for receiving requests for privileged processes' (emphasis added). In addition, Kouznetsov discloses a "configuration component [which] specifies a list of installable code components that are authorized for installation, wherein the agent will only execute privileged-mode functions in response to accesses by the user-mode code component when the installable code component is represented on the list" (emphasis added).

However, disclosing an API with a configuration component, as in Kouznetsov, simply fails to meet applicant's claimed technique "wherein said execution process maps configuration data specified within said operation specifying XML data to a configuration data store of said destination computer" (emphasis added), in the context claimed. In addition, Kouznetsov's disclosure of "[a] configuration component specif[ying] a list of installable code components" (emphasis added) in no way meets a technique "wherein said configuration data store is one of: a Windows Registry entry; an INI file; a DAPI store; and a database entry" (emphasis added), as claimed by applicant.

In the Office Action dated 09/22/2006, the Examiner merely reiterated the Examiner's previous rejection of applicant's claimed language. Again, applicant respectfully notes that none of the language cited by the Examiner teaches a technique "wherein said execution process maps configuration data specified within said operation specifying XML data to a configuration data store of said destination computer; wherein

said configuration data store is one of: a Windows Registry entry; an INI file; a DAPI store; and a database entry," as specifically claimed by applicant.

Additionally, with respect to the independent claims, the Examiner has relied on the following excerpt from the Uszok reference to make a prior art showing of applicant's claimed technique "wherein an identifier of an execution process within said complex data type includes at least one of: data specifying a computer file to trigger said execution process; data specifying a communication channel to trigger said execution process; and data specifying an operating system command to trigger said execution process" (see this or similar, but not necessarily identical language in the independent claims).

> "[0068] Referring again to FIG. 5, establishing a botBox is initiated by the user or prompted by botMaster through the GUI 520. The GUI communicates via the botMaster core 530 to a botBox proxy component 542. **The botBox proxy component initializes local botBox storage 536 through the botMaster configuration component 534 and then utilizes a communication manager 544, and more specifically a botBox communicator component 546, to send a message to request initialization of a corresponding botBox on a botServer.** Any botServer (with botBox facilities) can be used. Preferably, the user will want to select a botServer with high availability if the user bot(s) need to operate around the clock. Bots with more modest requirements can have their botBoxes hosted at more modest botServers, or even on their own home PC. The user can relocate botBox later if desired, and could store it (including all bot information) on machine-readable media for subsequent uploading on another platform. For commercial applications, botBoxes should be housed on a reliable server, preferably independent of the user platform. This architecture supports operation of the user's bots while the user is off-line, and ensures the user's privacy." (Uszok, Paragraph 0068 – emphasis added)

Applicant respectfully asserts that the above excerpt from Uszok relied upon by the Examiner discloses that "[t]he botBox proxy component initializes local botBox storage 536 through the botMaster configuration component 534 and then utilizes a communication manager 544 ... to send a message to request initialization of a corresponding botBox on a botServer" (emphasis added). However, "send[ing] a message to request initialization of a corresponding botBox," as in Uszok, fails to meet a technique "wherein an identifier of an execution process within said complex data type includes at least one of: data specifying a computer file to trigger said execution process;

data specifying a communication channel to trigger said execution process; and data specifying an operating system command to trigger said execution process" (emphasis added), as claimed by applicant.

In the Office Action dated 09/22/2006, the Examiner relied on the following excerpt from the Uszok reference to support the above rejection.

> "[0054] Importantly, each bot may consist (and usually does) of two or more components, for example, an sBot component 312 and a corresponding mBot component 314. The present invention is not limited, however, to bots having a one-to-one relation between a single mBot and a single sibling sBot. For example, in some applications, one "administrator mBot" might control several utility sBots that provide monitoring services, translate messages, etc. Also, an sBot can clone itself to create a multiple-sBot team, as described later with regard to FIG. 9. For ease of explanation, this description focuses primarily on a one-mBot to one-sBot scenario, although extensions (cloning) to multiple sBots are mentioned later. **Each portion of the bot, mBot and sBot, is implemented as a separate executable program designed to communicate and interact with its counterpart.** We call this bifurcation of the bot code. **The mBot portion executes exclusively on the end user's (client) machine, while the corresponding sBot portion executes on a server platform,** hereinafter called a "botServer" (350). To illustrate, botServer is hosting sBots 352 and 354. The two parts of the bot need not necessarily--and frequently do not--execute concurrently. Each sBot and the corresponding (sibling) mBot have knowledge of their sibling common globally unique identifier. The mBot and sBot communicate via the botBox server 360, as explained in detail later." (emphasis added)
>
> "[0064] Preliminarily, **the botMaster program itself is downloaded or otherwise installed on the user platform. To initialize the botMaster, the user interacts via the GUI 520 with the botMaster core 530, which in turn communicates with a botMaster configuration component 534.** A multi-user manager 532 can be implemented to support more than one user. Each user will have its own setup data, botBox, local storage 540, etc. In botMaster configuration, user setup data 538 is stored and botBox storage 536 is initialized. A knowledge manager component 590 can immediately begin developing the User Model (592)." (emphasis added)

Applicant respectfully points out that the above excerpts merely disclose that "[e]ach portion of the bot, mBot and sBot, is implemented as a separate executable program" and that "[t]he mBot portion executes exclusively on the end user's (client)

machine." In addition, Uszok teaches that "[t]o initialize the botMaster, the user interacts via the GUI 520 with the botMaster core 530, which in turn communicates with a botMaster configuration component."

However, implementing mBot as "a separate executable program" and disclosing that botMaster is initialized when "the user interacts via the GUI 520 with the botMaster core," as in Uszok, simply fails to teach "an identifier of an execution process within said complex data type," much less a technique "wherein an identifier of an execution process within said complex data type includes at least one of: data specifying a computer file to trigger said execution process; data specifying a communication channel to trigger said execution process; and data specifying an operating system command to trigger said execution process" (emphasis added), as claimed by applicant.

With respect to the independent claims, the Examiner has relied on the following excerpt from the Uszok reference to make a prior art showing of applicant's claimed technique "wherein said result data includes data specifying existing configuration data of said destination computer" (see this or similar, but not necessarily identical language in the independent claims).

> "[0122] **The user may choose to create the dynamically created, temporary profiles based on existing ones.** The temporary profile is used for a limited amount of time (specified by timeout, response type, or any other kind of trigger) or per-transaction and after that it is destroyed. This allows the user to block all the later communication attempts (especially spam) with a particular profile--they will fail because the profile does not exist anymore. It also supports the anonymity in the bot system--it is difficult (if possible at all) to track a person using the dynamic profile. The user may also choose to define the information filters on each user profile. Any information flow going through the profile may be analyzed (e.g., checking the sender, analyzing the message content, etc.) and rejected if does not fit to the rules defined in the filter. This allows an additional anti-spam mechanism to be built into the user profiles. The rejected information is not routed to (and from) the user account (optionally only notifying it about the information rejection)" (Uszok, Paragraph 0122 - emphasis added)

Applicant respectfully asserts that the excerpt from Uszok above relied upon by the Examiner merely teaches that "[t]he user may choose to create the dynamically created, temporary profiles based on existing ones." However, there simply is no mention in the above excerpt of a technique "wherein said result data includes data specifying existing configuration data of said destination computer" (emphasis added), as claimed by applicant.

In the Office Action dated 09/22/2006, the Examiner responded by stating that "[i]n the mBot architecture, the configuration and task results presentation block is genereated (0110) and the mBot can send the result back to the sBot (0111)" and "[t]herefore, Uszok in view of Kouznetsov discloses the limitations in the claims and the rejection of the claims are maintained." Additionally, the Examiner cited the following excerpts from Uszok to support such assertions.

> "[0110] FIG. 7 provides a simplified, general overview of an mBot architecture. The mBot is also an executable application program, designed to interact with one or more related sBot siblings, although in most cases interactions will take place via the botBox intermediary. (For some applications, such as those requiring near real time results, direct communication between mBot and sBot can be effected using known technologies and protocols.) An mBot architecture is centered around an mBot logic block in a configuration similar to that of the sBot architecture of FIG. 6. **An mBot 700 is, of course, hosted by a botMaster application 704. A user control block 706 of the botMaster (coupled to the GUI of FIG. 5) allows the user to show or hide the mBot "skin."** The user control block communicates with a skin engine 720. **The user can invoke a predetermined configuration for the presentation interface, again via the user control block, utilizing a "configuration and task results presentation block" 722. The mBot displays messages, data and other results via the results presentation block 722,** in communication with a botMaster presentation space service 724. The mBot utilizes a bot data storage block 710 for storing all necessary data, under control of an mBot logic block 702. The mBot automatically stores and loads its data in the botMaster data store, utilizing a botMaster storage service 712." (emphasis added)

> "[0111] The configuration block 722 can be used for selecting or configuring a required logical function to be carried out by the logic block 702. **For example, the configuration block 722 can receive parameters from the user control block, incorporate them and provide them in a call to the logic block 702.** The mBot can also send messages to the sBot by utilizing a send-message block

740. All messages originating in the send-message block are transmitted to the sBot via a botMaster's message service 742. BotMaster messaging services were described earlier with reference to FIG. 5. Conversely, all messages received from the sBot (via the botMaster message service) are directed to a message dispatcher block 730 in the mBot for routing or delivery as appropriate. For example, some messages will be routed to the logic block while others might invoke methods in a dispatcher block 730, for example, to store data in the data block 710 directly." (emphasis added)

Applicant respectfully notes that the above excerpts relied on by the Examiner merely teach that '[t]he user can invoke a predetermined configuration for the presentation interface... via the user control block... utilizing a "configuration and task results presentation block" 722' and that "[t]he mBot displays messages, data and other results via the results presentation block 722" (Paragraph 0110). Further, Uszok discloses that "the configuration block 722 can receive parameters from the user control block, incorporate them and provide them in a call to the logic block 702" (Paragraph 0111). However, displaying "messages, data and other results via the results presentation block" (emphasis added), and "receiv[ing] parameters from the user control block, incorporat[ing] them and provid[ing] them in a call to the logic block" (emphasis added), as in Uszok, clearly does not teach a technique "wherein said result data includes data specifying existing configuration data of said destination computer" (emphasis added), as claimed by applicant. Clearly, the mere disclosure of receiving parameters from a user control block, as in Uszok, simply fails to meet "result data," in the manner as claimed by applicant. Further, the disclosure of a user showing or hiding the skin of a botMaster hosted mBot, as in Uszok, fails to suggest "result data includes data specifying existing configuration data of **said destination computer**" (emphasis added), as claimed by applicant.

Further, in the Office Action dated 09/22/2006, the Examiner additionally relied on paragraphs [0054] and [0064] from the Uszok reference (reproduced above) to support their above rejection. As previously mentioned, applicant respectfully points out that the above excerpts merely disclose that "[e]ach portion of the bot, mBot and sBot, is implemented as a separate executable program" and that "[t]he mBot portion executes exclusively on the end user's (client) machine" (paragraph [0054]) Further, Uszok

teaches that "[t]o initialize the botMaster, the user interacts via the GUI 520 with the botMaster core 530, which in turn communicates with a botMaster configuration component" and that "[i]n botMaster configuration, user setup data 538 is stored" (paragraph [0064] – emphasis added).

However, disclosing that a botMaster core "communicates with a botMaster configuration component" (emphasis added), as in Uszok, simply does not teach "result data," much less a technique "wherein said result data includes data specifying existing configuration data of said destination computer" (emphasis added), as claimed by applicant.

With respect to the independent claims, the Examiner has relied on the following excerpt from the Uszok reference to make a prior art showing of applicant's claimed technique "wherein said execution process maps existing configuration data of said destination computer stored within said configuration data store of said destination computer to said result data to be returned to said source computer" (see this or similar, but not necessarily identical language in the independent claims).

> "[0100] FIG. 11-B illustrates a master-slave arrangement for sharing a botBox. Here, the user installs a botMaster application 1118 on a portable device such as a PDA, cell phone, or automobile Internet appliance. This may be a "thin client" with a simplified user interface (perhaps with limited graphics), **and it may lack the software and memory necessary for creating and maintaining a user model.** This botMaster 1118 need not establish its own botBox account. Rather, the "thin" botMaster can interact with the same botBox 1112 that was established by the office botMaster 1110. **The portable botMaster, in a slave mode, employs the user model maintained by the master botMaster 1110 and the portable program assigns to its mBots one of the user profiles made available on the shared botBox 1112.** Creating new profiles or modifying existing ones can be done using the more full-featured office botMaster 1110." (Uszok, Paragraph 0100 - emphasis added)

Applicant respectfully asserts that the excerpt from Uszok above relied upon by the Examiner teaches that "[t]he portable botMaster, in a slave mode, employs the user model maintained by the master botMaster 1110 and the portable program assigns to its

mBots one of the user profiles made available on the shared botBox 1112" (emphasis added) since "it may lack the software and memory necessary for creating and maintaining a user model" (emphasis added). The disclosure of "employ[ing] the user model maintained by the master botMaster," as in Uszok, simply fails to even suggest a technique "wherein said execution process maps existing configuration data of said destination computer stored within said configuration data store of said destination computer to said result data to be returned to said source computer" (emphasis added), as claimed by applicant.

In the Office Action dated 09/22/2006, the Examiner responded by arguing that "[i]n the mBot architecture, the configuration and task results presentation block is genereated (0110) and the mBot can send the result back to the sBot (0111)."

> "[0111] The configuration block 722 can be used for selecting or configuring a required logical function to be carried out by the logic block 702. For example, the configuration block 722 can receive parameters from the user control block, incorporate them and provide them in a call to the logic block 702. **The mBot can also send messages to the sBot by utilizing a send-message block 740.** All messages originating in the send-message block are transmitted to the sBot via a botMaster's message service 742. BotMaster messaging services were described earlier with reference to FIG. 5. Conversely, all messages received from the sBot (via the botMaster message service) are directed to a message dispatcher block 730 in the mBot for routing or delivery as appropriate. For example, some messages will be routed to the logic block while others might invoke methods in a dispatcher block 730, for example, to store data in the data block 710 directly." (emphasis added)

Applicant respectfully points out that the above excerpts merely disclose that "[t]he mBot can also send messages to the sBot by utilizing a send-message block" (emphasis added) in addition to teaching that '[t]he user can invoke a predetermined configuration for the presentation interface... via the user control block... utilizing a "configuration and task results presentation block" 722' and that "[t]he mBot displays messages, data and other results via the results presentation block 722" (paragraph [0110]) as mentioned above. However, sending messages from the mBot to the sBot, in addition to displaying messages utilizing a "configuration and task results presentation block" fails to teach a technique "wherein said execution process maps existing

configuration data of said destination computer stored within said configuration data store of said destination computer to said result data to be returned to said source computer" (emphasis added), as claimed by applicant.

In the Office Action dated 09/22/2006, the Examiner additionally relied on paragraphs [0054] and [0064] from the Uszok reference (reproduced above) to support their above rejection. As previously mentioned, applicant respectfully points out that the above excerpts merely disclose that "[e]ach portion of the bot, mBot and sBot, is implemented as a separate executable program" and that "[t]he mBot portion executes exclusively on the end user's (client) machine" (paragraph [0054]) Further, Uszok teaches that "[t]o initialize the botMaster, the user interacts via the GUI 520 with the botMaster core 530, which in turn communicates with a botMaster configuration component" and that "[i]n botMaster configuration, user setup data 538 is stored" (paragraph [0064] – emphasis added). However, the mere disclosure that the user interacts via the GUI with the botMaster core which communicates with a botMaster configuration component, as in Uszok, simply fails to even suggest a technique "wherein said execution process maps existing configuration data of said destination computer stored within said configuration data store of said destination computer to said result data to be returned to said source computer" (emphasis added), as claimed by applicant.

With respect to the independent claims, the Examiner has relied on paragraphs [0054] and [0064] from the Uszok reference (reproduced above) to make a prior art showing of applicant's claimed technique "wherein said operation specifying XML data is parsed after validating said operation specifying XML data to extract at least one identifier for mapping said at least one identifier to an available execution process" and a technique "wherein said operation specifying XML data includes parameter data used by said execution process in said operation" (see this or similar, but not necessarily identical language in the independent claims).

As previously mentioned, applicant respectfully points out that the above excerpts merely disclose that "[e]ach portion of the bot, mBot and sBot, is implemented as a

separate executable program" and that "[t]he mBot portion executes exclusively on the end user's (client) machine" (paragraph [0054]) Further, Uszok discloses that "[t]o initialize the botMaster, the user interacts via the GUI 520 with the botMaster core 530, which in turn communicates with a botMaster configuration component" and that "[i]n botMaster configuration, user setup data 538 is stored" (paragraph [0064] – emphasis added). However, mere disclosure that the user interacts via the GUI with the boMaster core with communicates with a botMaster configuration component, as in Uszok, simply fails to even suggest a technique "wherein said operation specifying XML data is parsed after validating said operation specifying XML data to extract at least one identifier for mapping said at least one identifier to an available execution process" and a technique "wherein said operation specifying XML data includes parameter data used by said execution process in said operation" (emphasis added), as claimed by applicant.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *In re Vaeck*,947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

Applicant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art references, when combined, fail to teach or suggest all of the claim limitations, as noted above. Nevertheless, despite such paramount deficiencies and in the spirit of expediting the prosecution of the present application, applicant has incorporated the subject matter of former Claim 91 into the independent claims.

With respect to the subject matter of former Claim 91 (now at least substantially incorporated into the independent claims), the Examiner has relied on the Paragraphs 0051, 0055, and 0056 (reproduced above) from the Uszok reference to make a prior art showing of applicant's claimed "validating said operation specifying XML data received at said destination computer against schema data, where said schema data is sent to said destination computer from said source computer at the same time as said operation specifying XML data" (see this or similar, but not necessarily identical language in the independent claims).

Applicant respectfully asserts that the excerpts from Uszok relied upon by the Examiner merely disclose that "[a] number of formal systems have been proposed for expressing knowledge in forms that are useful for computers to exchange and acquire knowledge" where "[s]ome of the leading protocols include KIF (Knowledge Interchange Format)" (Paragraph 0051). Further, Uszok discloses that "[w]ithin this environment, bots can acquire and exchange knowledge using virtually any desired language, metalanguage or ontology" (Paragraph 0051). In addition, Uszok discloses that "[e]ach portion of the bot, mBot and sBot, is implemented as a separate executable program designed to communicate and interact with its counterpart" where '[t]he mBot portion executes exclusively on the end user's (client) machine, while the corresponding sBot portion executes on a server platform, hereinafter called a "botServer" (350)' and "[t]he mBot and sBot communicate via the botBox server 360" (Paragraph 0054 – emphasis added). In addition, Uszok discloses that "[w]hen the user deploys the bot…a launch message is sent from botMaster to the server (more precisely--to botBox) to launch the corresponding sBot" (Paragraph 0056 – emphasis added).

However, the mere disclosure of using a KIF format for computers to exchange information, that the mBot executes on the client machine, and the sBot executes on the botServer, where a user deployed bot sends a launch message to the botServer to launch the sBot, as in Uszok, simply fails to suggest "validating said operation specifying XML data received at said destination computer against schema data, where said schema data is sent to said destination computer from said source computer at the same time as said

operation specifying XML data" (emphasis added), as claimed by applicant. Clearly, the disclosure that computers communicate via KIF, as in Uszok, fails to suggest that "said schema data is sent to said destination computer from said source computer at the same time as said operation specifying XML data" (emphasis added), in the manner as claimed by applicant

Further, applicant has considered the Uszok reference in it's entirety, and respectfully asserts that paragraph 0128 in Uszok merely discloses that "[a] meeting place in the present system is a separated logical area of the botServer, where bots can interact with chosen types of other bots and plug-ins according to defined interaction protocols" (Paragraph 0128 – emphasis added). Further, Uszok discloses that "[i]nteraction protocols define a set of states that a bot may exist in, rules of transition from one state to another and [optionally] a description of the structure (schema) of documents (data) that can be exchanged between participating parties in order to transition from one state to another" where "[a]n interaction protocol can be described, for example, in an XML-based language" (Paragraph 0128 – emphasis added).

However, the mere disclosure of a separated logical area of the botServer where bots interact according to defined interaction protocols with may optionally define a schema of documents that may be exchanged between participating parties, as in Uszok, simply fails to suggest that "said schema data is sent to said destination computer from said source computer at the same time as said operation specifying XML data" (emphasis added), in the manner as claimed by applicant. Clearly, Uszok's interaction protocols occur in a separated logical area of the botServer, which fails to suggest that "said schema data is **sent to** said destination computer **from** said source computer" (emphasis added), in the manner as claimed by applicant.

Additionally, applicant has considered the Uszok reference in it's entirety, and applicant respectfully asserts that paragraph 0129 in Uszok merely discloses that "[a] bot developer kit (SDK) can be arranged to generate bot code to implement any of a selection of predetermined protocols" (Paragraph 0129 – emphasis added). Further, Uszok

discloses that "[t]hrough this implementation of protocols and standard data schema, interoperability of bots is enabled over a wide variety of practical applications" where "[a] Bot platform (botServer/botExecutor) ...excludes from interaction those bots that do not conform to the agreed protocol" (Paragraph 0129 – emphasis added).

Clearly, Uszok's disclosure of using a selection of predetermined protocols and standard data schema to enable interoperability of bots and to exclude bots that do not conform to the agreed protocol simply fails to even suggest that "said schema data is sent to said destination computer from said source computer at the same time as said operation specifying XML data" (emphasis added), in the manner as claimed by applicant. Furthermore, Uszok's teachings of using a standard schema to enable interoperability of bots actually *teaches away* from applicant's claimed sending "said schema data ...to said destination computer from said source computer at the same time as said operation specifying XML data" (emphasis added), in the manner as claimed by applicant, since the bots would already be utilizing the standard schema to ensure interoperability among the bots.

Again, applicant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art references, when combined, fail to teach or suggest all of the claim limitations, as noted above.

Thus, a notice of allowance or specific prior art showing of each of the foregoing claim elements, in combination with the remaining claimed features, is respectfully requested.

Still yet, applicant brings to the Examiner's attention the subject matter of new Claims 95-98 below, which are added for full consideration:

> "wherein said validating of said operation specifying XML data and said schema data transmitted from said source computer to said destination computer at the same time generates a validation result" (see Claim 95);

"wherein said validation result triggers at least one of a valid configuration response and an invalid configuration response" (see Claim 96);

"wherein said invalid configuration response generates an error message" (see Claim 97); and

"wherein said valid configuration response starts execution of an associated computer program" (see Claim 98).

Thus, all of the independent claims are deemed allowable. Moreover, the remaining dependent claims are further deemed allowable, in view of their dependence on such independent claims.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 505-5100. The Commissioner is authorized to charge any additional fees or credit any overpayment to Deposit Account No. 50-1351 (Order No. NAI1P480/01.298.01).

Respectfully submitted,
Zilka-Kotab, PC.

/KEVINZILKA/

Kevin J. Zilka
Registration No. 41,429

P.O. Box 721120
San Jose, CA 95172-1120
408-505-5100